UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Graduado en Ingeniería Informática

Interconexión de sistemas a través de un gestor de políticas de seguridad

Interconnection of systems through a security policy manager

Realizado por

José María Santos López

Tutorizado y cotutorizado por

María Cristina Alcaraz Tello y

Francisco Javier López Muñoz

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2019

Fecha defensa: julio 2019

Fdo. El/la Secretario/a del Tribunal

Resumen

El objetivo del TFG es implementar un sistema gestor de políticas de seguridad que controle el intercambio de información entre dispositivos, de forma que se mantenga la seguridad y la transparencia en las comunicaciones. El sistema está desarrollado en Python3, utilizando fundamentalmente las librerías Pycryptodome (Criptografía), Pyknow (Sistemas expertos) y PyMongo (Bases de datos no relacionales).

Este proyecto se diseña con el objetivo de ser un sistema centralizado para un mundo en el que los dispositivos del *Internet of Things* (IoT - en español: *Internet de las cosas*) están cada vez más presentes en el mundo desarrollado, y en el que es necesario una mejora en la seguridad que mantenga la confidencialidad y la integridad de las comunicaciones.

Con el fin de incluir el mayor número de dispositivos capaces de interactuar con este gestor, se han considerado distintos tipos según su capacidad de cómputo criptográfico. Para esto, el gestor incorpora un sistema experto que es capaz de determinar qué acciones realizar en función del tipo de dispositivo que envía y del dispositivo que recibe el mensaje. Además, se han tenido en cuenta los distintos roles que puedan existir en este entorno, como puede ser el de un administrador del sistema, un ingeniero que pueda comprobar los registros de las comunicaciones, o el de un usuario convencional que solamente utilice el sistema para enviar información.

Palabras clave: Internet de las Cosas, Redes de comunicación, Políticas de seguridad, Criptografía.

Abstract

The main objective of this Final Year's Project is to develop and implement a security policy manager that takes the role of controller between two devices, when they are engaged in a communication process, in order to maintain the security and transparency of said communications. The system is written using the programming language Python3, using mainly Pycryptodome (Cryptography), PyKnow(Expert systems) and PyMongo (non-relational DataBases) as auxiliary libraries.

This project is designed with the goal of being a centralized system in a scenario where IoT (Internet of Things) devices become more present in the developed world. There is a necessity in hardening security keeping confidentiality and integrity high in communications.

Finally, in order to include a wider range of devices capable of using this manager, it has been considered several types depending on their computing power when using cryptography tools. To achieve this, the manager incorporates an expert system that is capable of choosing the right action when a certain type of sender and receiver nodes are interacting with the system. In addition, it has been taken into consideration the variety of roles that could exist in this environment, as it could be an administrator, an engineer who could check the communication logs, or a simple user only capable of using the system to send a message.

Keywords: Internet of Things, Communication Networks, Secutity Policies, Cryptography.

Índice general

Introducción

Este es un Trabajo Fin de Grado (TFG) que ha sido desarrollado en la línea Seguridad de la información y las comunicaciones. En esta línea se busca la creación de un sistema que permita interconectar diferentes dispositivos que se encuentren dentro de la misma red, sean del tipo que sean, y utilicen las políticas de seguridad que mejor se adecuen a ellos. Con esto se consigue que la transmisión de información sea lo más transparente posible, sin prescindir del nivel de seguridad que estas comunicaciones requieren.

Al tratarse de un sistema intermediario en el que se pretende incorporar el uso de dispositivos IoT, móviles, e incluso ordenadores personales, se ha desarrollado para la realización de este TFG, además, una aplicación móvil para Android utilizando la herramienta Android Studio, que junto a un microcontrolador Arduino servirán para demostrar el funcionamiento del gestor de políticas en un entorno real.

Supongamos una red que dispone de diferentes dispositivos, y es necesario que se comuniquen entre ellos para el correcto funcionamiento de dicha red, ya que podría tratarse de una planta de producción automatizada en el que cada parte del sistema cubre un rol fundamental. Si la información que se transmiten entre ellos es crítica, la necesidad de ocultar esta información para evitar daños en la planta por robo de información es fundamental. Por ello, existe la posibilidad de que no todos los dispositivos utilicen políticas de seguridad que sean compatibles con el resto de la red. Con la incorporación del Gestor propuesto en este TFG, se posibilitaría la comunicación entre ellos aunque sus políticas de seguridad no sean compatibles, ya sea por no estar configurados para ello, o por su incapacidad de realizar los cómputos necesarios para recibir o enviar cierta información.

Para el desarrollo de este sistema, se han utilizado las librerías de Python Pycryptodome, PyKnow y PyMongo, entre otras. En los sucesivos capítulos se desarrollarán en profundidad las funcionalidades que estas librerías aportan. Además, para el

correcto funcionamiento del sistema, se ha incluido una base de datos no relacional basada en MongoDB, debidamente cifrada con los datos necesarios para registrar los distintos dispositivos que accederán al gestor, sus claves públicas, y un registro en el que se almacenarán datos relativos a las comunicaciones realizadas. Para el desarrollo de la aplicación móvil para Android, se ha utilizado un modelo cliente-servidor TCP para el envío de mensajes al gestor.

En las posteriores secciones se desarrollarán todos los aspectos técnicos y teóricos relativos de este TFG. La memoria está dividida en capítulos, y estos a su vez en apartados, los cuales se describirán a continuación: El capítulo 1, de introducción, explica las tecnologías utilizadas y el estado del arte, así como la motivación del proyecto. El capítulo 2 ofrece una visión general de la metodología utilizada en el desarrollo del proyecto seguida de un análisis de requisitos (capítulo 3). En el capítulo 4 se describe el diseño del gestor de forma general, mostrando los elementos fundamentales que lo hacen funcionar. El capítulo 5 entra en detalle el desarrollo propio del gestor de políticas, aportando los argumentos y aspectos técnicos del mismo. El capítulo 6 detalla la definición del sistema experto utilizado en el proyecto. Los capítulos 7 y 8 se centran en el uso de la aplicación Android y del microcontrolador Arduino y cómo se han llegado a utilizar en la demostración de pruebas y resultados (capítulo 9). Finalmente, se listan una serie de posibles mejoras aplicables al proyecto realizado en el capítulo 10, seguidas de las conclusiones finales (capítulo 11). Como anexo, se ha incluido el manual de usuario que permite su uso.

1.1. Tecnologías utilizadas

En esta sección se describirán brevemente las tecnologías utilizadas, así como las librerías que han hecho posible la realización de este proyecto. Se ha buscado en todo momento realizar soluciones con software libre.

- Python 3: debido a la gran versatilidad de este lenguaje de programación en el mundo de la seguridad de la información, se ha elegido para ser el elemento

principal en el desarrollo del proyecto.

- PyMongo: librería Python destinada para la inclusión de bases de datos no relacionales.

- Pycryptodome: librería Python utilizada para el uso de criptografía. Es la versión actualizada de la librería Crypto, que actualmente se encuentra obsoleta.

- PyKnow: librería Python para la implementación de un sistema experto basado en reglas.

- ECIES: librería destinada a la criptografía de curvas elípticas.

- Visual Studio: editor de código de Microsoft que incorpora una gran variedad de herramientas para el desarrollo de código.

- Android Studio: IDE de Google destinado al desarrollo de aplicaciones para dispositivos Android.

- Arduino: microcontrolador *open-source* de fácil uso para poder enlazar software y hardware.

- Java: para el desarrollo de la aplicación Android ha sido necesario el uso de este lenguaje de programación.
- C: para la comunicación vía serial entre la máquina anfitrión y el microcontrolador Arduino se ha utilizado el lenguaje de programación de bajo nivel C.

1.2. Estado del arte

En la actualidad se han diseñado y estudiado sistemas que se adecuen a la necesidad que se cubre en este proyecto. Sin embargo, a medida que aumenta el número de dispositivos IoT en el mercado, es de gran importancia que estos dispositivos se beneficien de una infraestructura segura en la que poder comunicarse sin restricciones.

1.2.1. Gestores de políticas

A día de hoy, en el mercado hay un número limitado de estos sistemas, y muchos de ellos se han destinado a soluciones personalizadas que cubren unas necesidades específicas que pueden no adaptarse a entornos no industrializados como es el hogar [4].

Algunos de estos gestores ofrecen soluciones como:

- Automatización de procesos de negocio.
- Unificar políticas de seguridad en todos los elementos del negocio, como puede ser la infraestructura *cloud* o el software dedicado a la integración del negocio a la red.
- Detección de vulnerabilidades.
- Factor de autenticación múltiple.
- Cifrado de información.
- Unificación de los equipos de desarrollo de la empresa.

Este tipo de soluciones, destinadas a entornos industrializados, no prestan atención a los dispositivos IoT mencionados anteriormente.

Un problema encontrado en este tipo de dispositivos, es su capacidad limitada de procesamiento. Esto se debe a que su diseño está orientado al uso en tareas fijas, sin necesidad de realizar cálculos computacionales demasiado elevados. Por ello, los fabricantes encuentran en estos dispositivos un reto que se debería cubrir cuanto antes [11]

La mayoría de directrices orientadas a la seguridad de la información son, a día de hoy, poco prometedoras en cuanto a su aplicación en dispositivos IoT. Esto es debido a que son destinadas a negocios, no al hogar [13].

Otro enfoque encontrado, más general que las anteriores soluciones, es estudiando el caso en el que los dispositivos toman un rol en una infraestructura determinada. Esta solución se acerca más a la mostrada en este proyecto, pero sigue sin tener en cuenta la fragilidad de estos dispositivos, los cuales pueden ser muy distintos entre sí debido a la no normalización de los modelos de diseño e implementación entre los fabricantes [10].

1.3. Motivación

La motivación por la cual se ha elegido este TFG es la de mejorar la seguridad dentro de las redes de dispositivos que cada día aumentan en número en entornos como el hogar, las empresas o el sector público. Por esto, es de gran importancia el uso de comunicaciones seguras. Si bien es cierto que algunos fabricantes compatibilizan sus dispositivos con políticas de seguridad, estas pueden no ser idénticas para aquellos de otros fabricantes. Es por este motivo que este TFG afronta este problema, hacer que todos los dispositivos de una misma red sean capaces de comunicarse entre ellos de forma transparente y sin necesidad de modificaciones en los mismos.

1.4. Objetivos del TFG

El objetivo principal de este TFG es mejorar la seguridad en las comunicaciones entre dispositivos, otorgándoles un sistema intermediario que permita realizarlas de forma transparente, sin necesidad de utilizar políticas de seguridad comunes entre

ellos. Para cumplir esto, se ha realizado una división de objetivos:

- Creación de un sistema experto que gestione el uso de cualquier tipo de dispositivo.
- Creación de un protocolo de comunicación que permita una transmisión segura de la información.
- Ofrecer un almacenamiento seguro de la información otorgada por los dispositivos.
- Ofrecer un mantenimiento en la integridad de la información, tanto en la comunicación como en el almacenamiento.

Para cumplir estos objetivos, se ha realizado un análisis de requisitos que se detallará en el capítulo 3.

Metodología

En este proyecto se siguió el modelo de desarrollo ágil Scrum [14]. Se van a detallar algunos aspectos clave de este modelo aplicado al proyecto:

- Se realizaron *sprints* cada dos semanas, en los cuales se alcanzaban metas progresivas y se realizaba un seguimiento del proyecto.

- Al final de cada sprint se definían claramente los pasos a seguir, teniendo en cuenta los requisitos aplicados al sistema a desarrollar.

- Estas reuniones se realizaban con los tutores, que harían de Scrum Master.

Los *sprints* se dividieron en diferentes fases: Diseño general del gestor de políticas, diseño y desarrollo del sistema experto, diseño del protocolo de comunicación, desarrollo del sistema, verificación y pruebas. Todas estas fases determinan el buen funcionamiento y diseño del proyecto. Una breve explicación de cada una de estas fases es la siguiente:

- Diseño general del gestor de políticas: siguiendo los requisitos y haciendo un estudio del estado del arte, se establecen los parámetros con los cuales el gestor se pondría en funcionamiento, y la estructura general del mismo.

- Diseño y desarrollo del sistema experto: según las necesidades del gestor, se analizaron como interactuarían los elementos que hacen funcionar al mismo, y se definieron así los atributos, reglas y acciones que componen el sistema experto de este TFG.

- Diseño del protocolo de comunicación: una vez definido el sistema experto, se propuso la implementación de un protocolo de comunicación propio para el gestor, indicando los *tokens* que componen los mensajes transmitidos, su

estructura y los requisitos necesarios previos al uso del gestor.

- Desarrollo del sistema: definidos y aclarados todos los apartados previos, se prosiguió a la implementación del código necesario para el funcionamiento y para las pruebas necesarias realizadas al final del mismo.

- Verificación y pruebas: Tras crear el código, se comprobó el buen funcionamiento del mismo desarrollando una pequeña aplicación Android y un microcontrolador Arduino.

En la figura 1 se muestra un diagrama de Gantt que detalla la linea temporal del desarrollo del proyecto. Un diagrama de Gantt es una herramienta que permite visualizar con facilidad el tiempo dedicado a cada tarea dentro de un proyecto. En este caso se ha dividido en 5 tareas principales, las cuales se han definido en la lista anterior.

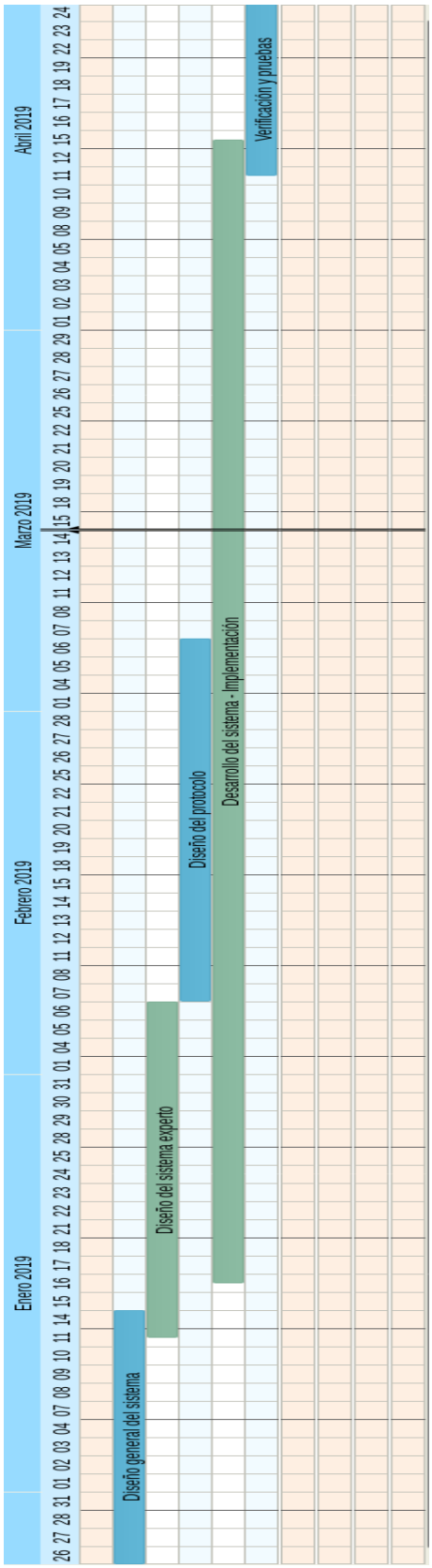

Figura 1. Diagrama de Gantt del desarrollo

Las 5 tareas se han ido desarrollando de forma incremental a lo largo del tiempo. En un primer lugar se realizó el diseño del gestor para aclarar los elementos clave que iban a formar parte del mismo. Finalizada esta fase, se comenzó el diseño del sistema experto, junto a parte de la implementación del gestor de políticas, ya que estas dos fases comenzaron simultáneamente. A medida que se avanza en el desarrollo del sistema, se diseña el protocolo de comunicación utilizado. Una vez finaliza este diseño, se siguió con el desarrollo a un primer plano, perfilando los elementos clave del sistema. Tras finalizar el desarrollo, se propusieron una batería de pruebas que verificaran el buen funcionamiento del mismo.

Más adelante se detallarán las fases de desarrollo.

Análisis de Requisitos

Durante las diversas reuniones con los tutores de este TFG, se analizaron una serie de requisitos que determinan la funcionalidad y la estructura del mismo. En este apartado se expondrán sus descripciones y en secciones posteriores se desarrollarán en detalle.

3.1. Requisitos funcionales

Aquí se definen aquellas funcionalidades necesarias para la correcta ejecución del proyecto. En esta sección (ver tabla 1) se muestra un listado de los principales requisitos funcionales recogidos tras las entrevistas.

Tabla 1: Requisitos funcionales

ID	Nombre	Descripción
RF01	Iniciar sesión	El nodo registrado podrá iniciar sesión al sistema con su contraseña generada previamente.
RF02	Modificar claves	El nodo podrá modificar sus claves ECC.
RF03	Enviar mensaje	El nodo podrá enviar un mensaje estructurado siguiendo el protocolo definido.
RF04	Ver registro	El nodo habilitado podrá ver el listado de operaciones realizadas por fecha.
RF05	Añadir usuario	El nodo habilitado podrá añadir al sistema un usuario aportando toda la información necesaria.
RF06	Eliminar usuario	El nodo habilitado podrá eliminar un usuario del sistema aportando su nombre de registro.

3.2. Requisitos no funcionales

Aquí se definen los atributos de calidad que hacen posible que el proyecto sea desarrollado cumpliendo los objetivos fijados en las distintas reuniones realizadas. En

esta sección (ver tabla 2) se muestran los principales requisitos no funcionales.

Tabla 2: Requisitos no funcionales

ID	Nombre	Descripción
RNF01	Protección de contraseñas	Las contraseñas estarán almacenadas como *hash*, aplicándole un salt.
RNF02	Protección de base de datos	La base de datos estará protegida utilizando un cifrado que únicamente el servidor podrá leer.
RNF03	Cifrado de mensajes	Todos los mensajes producidos estarán cifrados con clave pública para que únicamente el nodo receptor pueda leerlos.
RNF04	Disponibilidad de datos	Los datos de la base de datos deberán poder ser consultados por un nodo habilitado en cualquier momento.

3.3. Actores

Los actores encargados de interactuar con el sistema son los siguientes:

- **Nodo habitual**: representa a todos los nodos que acceden al sistema. Deben estar registrados y accederán al mismo mediante una contraseña (ver figura 3).

- **Administrador**: este actor se encargará de todas las operaciones de mantenimiento y gestión de la base de datos, así como del sistema en general. Es el tipo de usuario que tiene los máximos privilegios del sistema. Controlará además la información relativa al servidor, ya sean sus claves o políticas utilizadas (ver figura 4).

- **Ingeniero**: este actor se encargará del control de los registros (ver figura 5).

- **Usuario final**: representa a todos los dispositivos que harán uso del sistema. Este actor podrá adquirir cualquier rol definido en el sistema (ver figura 2).

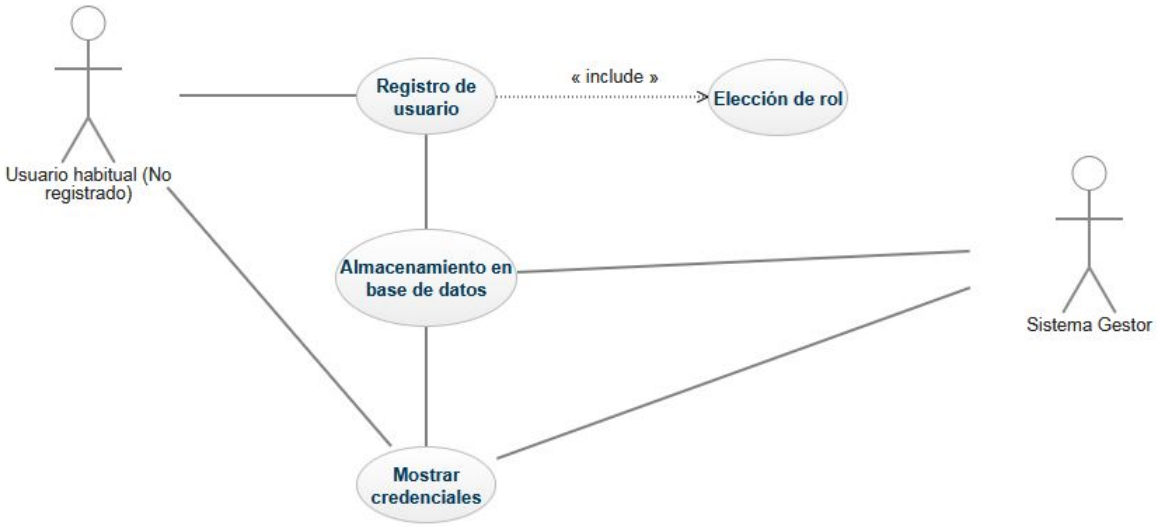

Figura 2. Caso de uso de registro de nuevo usuario

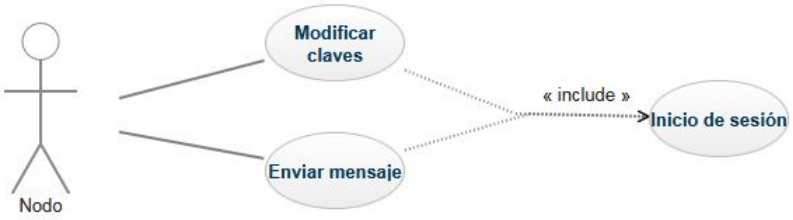

Figura 3. Caso de uso de nodo

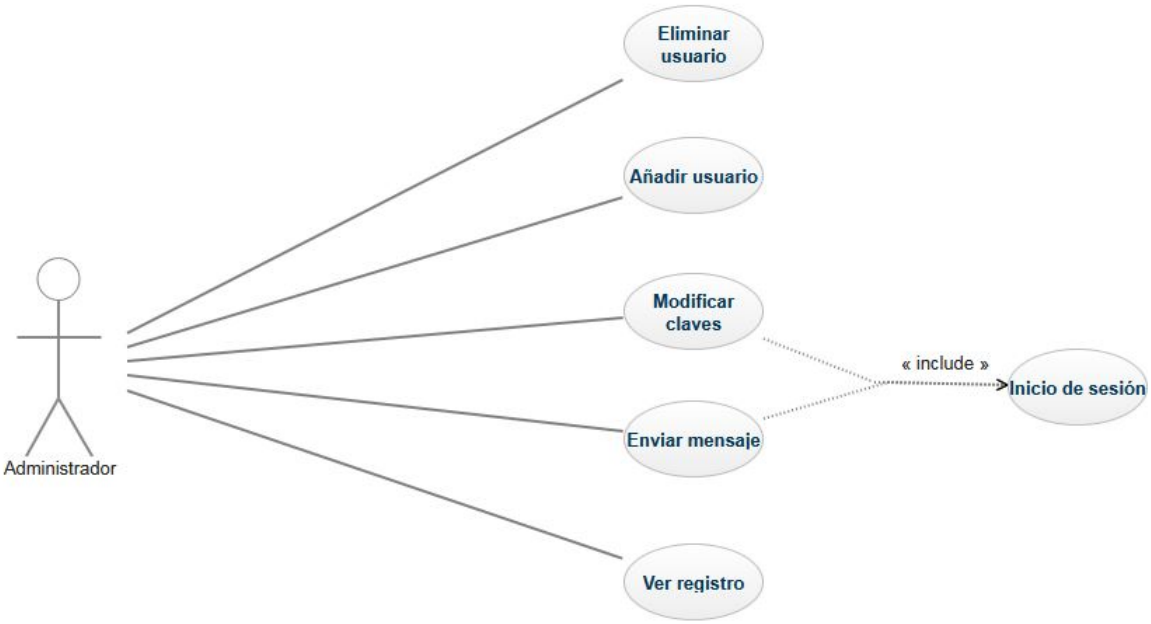

Figura 4. Caso de uso de administrador

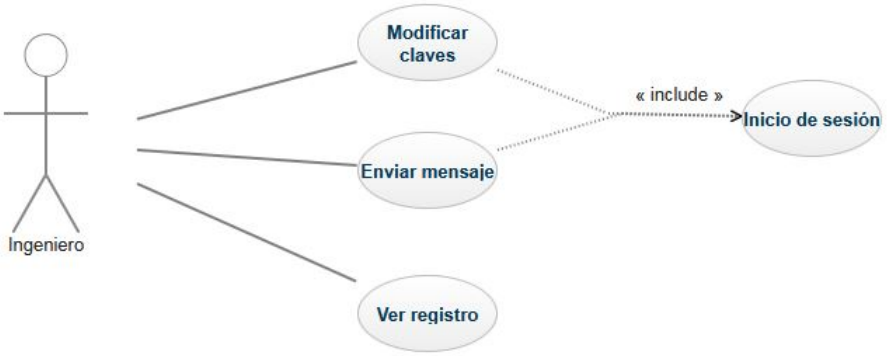

Figura 5. Caso de uso de ingeniero

Diseño del Gestor

El gestor se compone de tres elementos principales, los cuales se comunican con el mundo real a través del sistema principal. Estos elementos son:

- Gestor de políticas: este agrupa todos los elementos que componen el sistema. Incorpora un sistema de autenticación por contraseña, a la cual se le aplica un hash y es generada automáticamente siguiendo los siguientes criterios:
  - 16 caracteres alfanuméricos
  - Se utiliza SHA512 para el *hash* de la contraseña, aplicando un *salt* para una mayor seguridad de la misma. Un *hash* es una función unidireccional gracias a la cual se puede verificar la validez de un dato. Un *salt* es un valor aleatorio que sirve para dificultar la obtención del valor original del dato al cual se le aplica el *hash*.
  - La contraseña se regenera cada 365 días, actualizándose así todos los campos necesarios para su correcta actualización [28].
- Base de datos: aquí se almacena toda la información necesaria para el correcto funcionamiento del sistema. En ella se integran tres tablas fundamentales, estas son:
  - **Nodes** (ver tabla 3): en esta tabla se almacenan todos los atributos relativos a cada nodo del entorno en el que está integrado el gestor.

Tabla 3: Atributos de la tabla Nodes

header	**significado**
_id	Identificador único de fila
ID	Nombre del nodo registrado en el sistema
Type	Tipo de nodo, según criterios elegidos (W, P, H)[1]
Cipher	Cifrados que el nodo es capaz de utilizar
LastAccPwdUpdate	Última fecha en la que se modificó la contraseña
Password	Hash de la contraseña + salt utilizado
Salt	salt utilizado en la contraseña del nodo
Role	Rol del nodo en el sistema (Eng, Admin, Node)[2]

[1] *(Weak, Powerful, Hard): son los tipos de nodos que se pueden registar en el gestor de políticas.*

[2] *(Eng, Admin, Node): roles definidos en el sistema. Admin tiene todos los privilegios, Eng posee los privilegios de gestión de registros, y Node será el rol más limitado.*

- **PubKeys** (ver tabla 4): en esta tabla se almacenan las claves públicas de los nodos registrados en el gestor. Además, se incluyen las claves utilizadas por el sistema.

Tabla 4: Atributos de la tabla PubKeys

header	**significado**
_id	Identificador único de fila
PubKey	Clave pública del nodo
PrivKey (Caso Servidor)	Clave privada del sistema

En el caso del almacenamiento de la clave privada del sistema, se ha establecido así por conveniencia del proyecto. En un entorno real, esta clave privada debería estar almacenada en un lugar seguro, inaccesible por terceras partes.

- **Logs** (ver tabla 5): en esta tabla se almacena un registro de comunicación sencillo que serviría para comprobar qué acciones se han realizado en una fecha determinada.

Tabla 5: Atributos de la tabla Logs

header	**significado**
_id	Identificador único de fila
Sender	Nodo que emite el mensaje
Receiver	Nodo que recibe el mensaje, modificado o no
Date	Fecha en la que se realizó dicha comunicación

- Sistema Experto: este módulo se encarga de seleccionar la acción correcta dependiendo de los tipos de nodos que accedan al sistema. Sigue un motor basado en reglas que sigue la estructura <**regla**> := <**condición**> $\Rightarrow$ <**acción**> tal que la <*condición*> contiene los predicados asociados al <$sujeto$><$objeto$><$mensaje$>, en este caso, nodos emisor y receptor. El sistema experto se ha dividido en dos fases, una inicial que verifica el rol del nodo que accede al sistema, y una final que determina qué acciones tomar frente al nodo receptor.

En el esquema representado en la figura 6 se resume el funcionamiento del gestor de políticas:

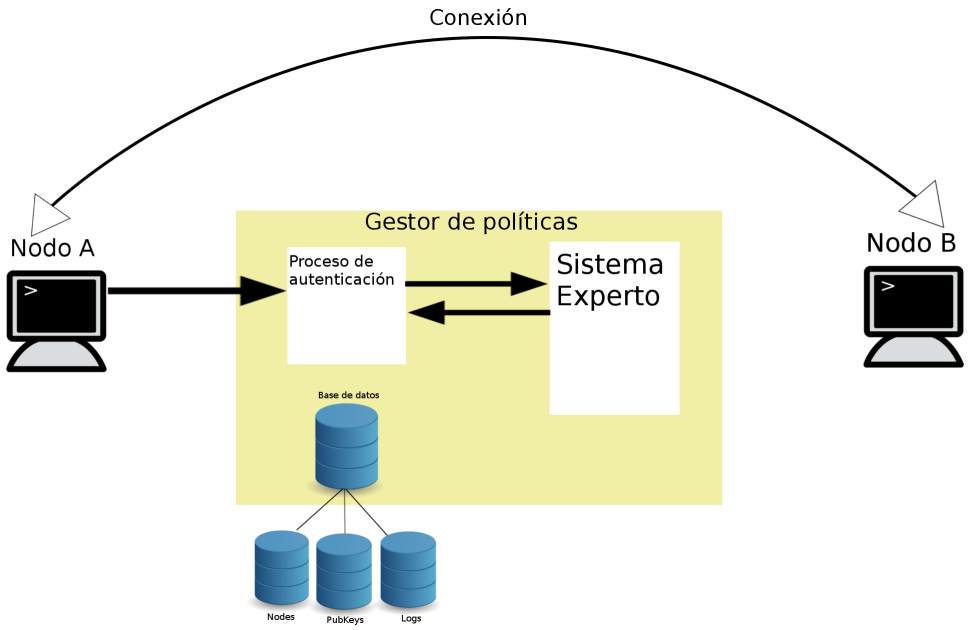

Figura 6. Esquema del gestor de políticas

En cuanto al funcionamiento del sistema, un dispositivo (denominado nodo A) desea enviar un mensaje a otro dispositivo (denominado nodo B). Para ello, accede al sistema a través de un proceso de autenticación que verifica su identidad obteniendo su información de la tabla **Nodes** de la base de datos. Cuando este accede, el gestor puede proceder a recibir el mensaje deseado para su procesamiento y modificación, gracias al sistema experto. Finalmente, este reenvía el mensaje modificado al Nodo A haciendo uso de su clave pública obtenida de la tabla **PubKeys** de la base de datos. Una vez el mensaje es recibido por el nodo A, el gestor archiva la comunicación que va a realizarse entre los nodos A y B. Es en este momento en el que el nodo A es capaz de enviarle su mensaje original de forma que el nodo B puede procesarlo (fase de conexión).

A continuación se va a explicar en detalle el desarrollo del proyecto, profundizando en cada fase del mismo, y mostrando las soluciones y el por qué de las mismas.

Desarrollo del Gestor de políticas

En este capítulo se expondrán todo lo relativo al desarrollo del gestor de políticas. Se incluirán los aspectos técnicos y lógicos que dan vida al sistema.

5.1. Autenticación

El primer paso para un correcto funcionamiento del mismo es la autenticación. En esta fase se ha implementado una ventana de inicio de sesión en la que el nodo que acceda al sistema requiera de la inserción de una contraseña. Esta contraseña se genera automáticamente en el proceso de registro, aleatorizando 16 caracteres alfanuméricos que serán mostrados al nodo una vez complete dicho proceso de registro.

En este proceso al usuario se le piden los siguientes parámetros:

- **Rol**: los roles disponibles en el registro son ingeniero (Eng) y nodo usual (Node) (ver tabla 7). El rol de administrador no es posible registrarlo haciendo un uso habitual del sistema, deberá ser incorporado en la base de datos manualmente por el administrador del gestor.

- **Cifrado utilizado**: este parámetro se ha determinado como dependiente de las capacidades del nodo que se va a registrar. El sistema detecta las características del procesador del mismo y en función de ellos determina qué cifrados puede utilizar. Esta solución se ha tomado así por conveniencia del proyecto. En un entorno real, este parámetro deberá ser cumplimentado en la fase de registro con los cifrados que dicho nodo sea capaz de utilizar.

El resto de parámetros son generados por el sistema por conveniencia en el desarrollo del gestor de este proyecto. Esto es así, debido a la búsqueda de transparencia a la hora del uso del mismo. Se intenta reducir al máximo la necesidad de la inserción

de información por parte de los nodos, ya que estos serán dispositivos autónomos que emiten y reciben información, como es el ejemplo de Amazon Echo [1].

Tras el registro del nodo, toda la información es almacenada en la base de datos, aplicándole un cifrado por ECC, Criptografía de Curvas Elípticas (Elliptic Curve Criptography en inglés) [31] utilizando las claves del sistema. A la contraseña se le aplica, además, un hash utilizando SHA512 y un salt. Debido al reciente descubrimiento de una vulnerabilidad encontrada en SHA1 [22], se ha decidido utilizar este algoritmo hash para una mejor protección de las contraseñas.

Cada vez que un nodo accede al sistema, se comprueba si dicha contraseña sigue siendo válida dependiendo de la fecha de su última actualización. Según el NIST, el Instituto Nacional de Estándares y Teconlogía (National Institute of Standards and Technology en inglés) [28], ya no se requiere de una actualización de la contraseña por parte del usuario. Según recientes estudios, la obligación por parte del usuario de mantener actualizadas sus contraseñas ha sido demostrado ser menos beneficioso de lo que se esperaba. Es por esto que Microsoft dejaría de obligar a sus usuarios a modificar sus contraseñas [24]. En el caso de este TFG, se ha decidido mantener una política de actualización de contraseñas con un periodo de tiempo de 365 días, ya que al ser un proceso transparente al usuario, se mantiene un nivel de seguridad constante.

5.2. Gestión de claves

De acuerdo con el mantenimiento de la integridad y confidencialidad de los datos, todos los mensajes que se transmitan de un usuario a otro por medio de este gestor han de ser cifrados con sus debidas claves públicas, utilizando la ya mencionada ECC. Se ha utilizado este estándar ya que es óptimo para el uso del mismo en dispositivos con baja potencia computacional, como son los dispositivos IoT. Además, el tamaño usado para sus claves es considerablemente menor que su contrapartida usando RSA (Rivest, Shamir y Adleman, nombre adquirido por sus autores), que es

uno de los sistemas criptográfico de clave pública más utilizado a en la actualidad [23], manteniendo un nivel de protección similar (ver tabla 6).

Tabla 6: Tamaño de claves recomendados (NIST)

Symmetric Key Size (bits)	**RSA and Diffie-Hellman Key Size (bits)**	**Elliptic Curve Key Size (bits)**
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

En cuanto a la criptografía de curvas elípticas, este cifrado se basa en los conceptos algebraicos que manipulan las curvas elípticas en el plano. Estas curvas tienen la propiedad de, si se unen dos puntos A y B, es posible construir un tercer punto C. Estos 3 puntos pertenecen a un grupo algebraico en el se puede derivar un punto A en otro C. La forma en la que se construye C a partir de A no es fácil de determinar, lo cual lo hace un método robusto de criptografía. Las características que hacen que la criptografía de curvas elípticas un método eficaz son:

- Sus claves son de tamaño reducido, lo cual facilita su almacenamiento y cómputo.
- Crear una nueva curva es costoso, lo cual producir un ataque al método es improbable.
- Las curvas elípticas pueden usarse para factorizar enteros.
- Algunas curvas elípticas pueden usarse para emparejar. El término emparejamiento se estudia para involucrar tres participantes en un protocolo. Por ejemplo en este caso sería incluir en la comunicación el servidor, el nodo emisor y el nodo receptor.

5.3. Cifrados

El objetivo del sistema es ser capaz de interconectar dispositivos, independientemente de las políticas de seguridad que utilicen o de la capacidad que tengan. Es por esto que se ha determinado un número de cifrados que el gestor es capaz de utilizar para que el conjunto del entorno funcione correctamente.

Se ha elegido como medio principal la criptografía simétrica. Este tipo de criptografía se caracteriza por hacer uso de una clave común a los dos extremos de la comunicación que utilizarían para enviar y recibir información entre ellos. El principal inconveniente del uso de esta criptografía es la dificultad de transmisión de la clave simétrica, siendo necesario auxiliarse de criptografía asimétrica para este fin. El listado de algoritmos de cifrado usados en este proyecto es el siguiente [6]:

- **AES**: *Advanced Encryption Standard* por sus siglas en inglés, es uno de los algoritmos más utilizados a día de hoy. Utiliza claves de 16, 24 o 32 bytes y, dependiendo del modo en el que se utilice, necesita de un vector de inicialización o de un nonce, de un tamaño recomendado de 16 bytes.

- **DES**: *Data Encryption Standard* por sus siglas, es uno de los algoritmos más conocidos. Aunque es considerado inseguro en la actualidad, se ha incluido en el proyecto para reflejar la posibilidad de que existan algunos dispositivos que aún utilicen este tipo de cifrado por no haber sido actualizados. Las clave que utiliza son de 8 bytes, y en el caso de que necesite vector de inicialización o nonce, también lo serán de 8 bytes de longitud.

- **3DES**: basado en el algoritmo DES, es una sucesión en cascada de tres cifrados DES, en los cuales cada uno de ellos emplea una sub-clave única e independiente. Tiene las mismas características que DES, pero triple DES es aún uno de los algoritmos recomendados en [19].

- **Blowfish**: es un algoritmo diseñado como reemplazo a DES y a IDEA. Es con-

siderado lento por parte de la comunidad en algunas aplicaciones y vulnerable a *Birthday Attacks*. Estos ataques abusan de los conceptos matemáticos que fundamentan el algoritmo [25]. Toma claves de entre 5 a 56 bytes. En el caso de necesitar vector de inicialización o nonce, estos serán de 8 bytes.

Junto a estos algoritmos de cifrado, se ha incluido la posibilidad de uso de dos de sus modos de operación, estos son:

- **Modo CBC**: es un modo de operación por bloques a los que se les aplica la operación lógica XOR junto con un vector de inicialización (IV - initialization vector) como acción previa al cifrado.

- **Modo EAX**: es un modo de cifrado autenticado el cual asegura la confidencialidad y la autenticidad del dato [3].

5.4. Mensaje

Para el funcionamiento del protocolo de comunicación de mensajes, se ha diseñado una estructura la cual deberá cumplirse para la correcta ejecución del gestor de políticas, como se muestra en la figura 2.

Este mensaje tiene los siguientes atributos:

<Cifrado usado>
<IV\Nonce usados>
<Clave usada>
<Contenido del mensaje cifrado>

Tal que:

- Cifrado usado, IV o Nonce y clave usada seguirán uno de los algoritmos y atributos de los descritos en la subsección superior.

- Contenido del mensaje cifrado será lo que se requiere que el nodo receptor procese en última instancia.

Como protección, el mensaje se cifra utilizando la clave pública del servidor (el gestor, en este caso), para su posterior procesamiento y transformación.

5.5. Transformación del mensaje

Una vez recibido el mensaje por parte del gestor, y haciendo uso del sistema experto, se pasa a la fase de adaptación del mismo.

Este mensaje, una vez transformado, toma una estructura similar a la establecida en la fase 1 del protocolo de la figura 7, junto a su diagrama de actividad en la figura 8, con la inclusión de un elemento de control de integridad que será el *hash* del mensaje cifrado que se incluye en el mismo. Gracias a este *hash*, se le permite al nodo receptor comprobar que su mensaje no ha sido modificado, y en el caso de no ser correcto, el nodo receptor no aceptaría su recepción. Para proteger este *hash*, se envuelve en un cifrado con su clave pública dentro del bloque cifrado previamente creado.

Por tanto, el mensaje tomaría los siguientes atributos:

<Cifrado nuevo>
<IV\Nonce nuevos>
<Clave nueva>
<Contenido del mensaje cifrado>
<Hash del mensaje cifrado, protegido con clave pública>

En la siguiente sección se detallará el desarrollo del sistema experto, explicando los casos posibles y las acciones a tomar por parte del gestor de políticas.

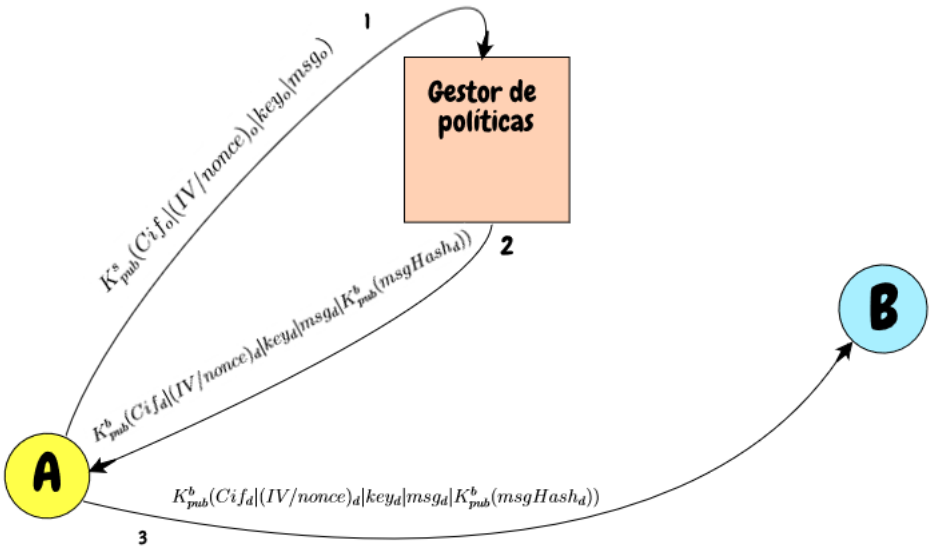

Figura 7. Protocolo de envío de mensajes

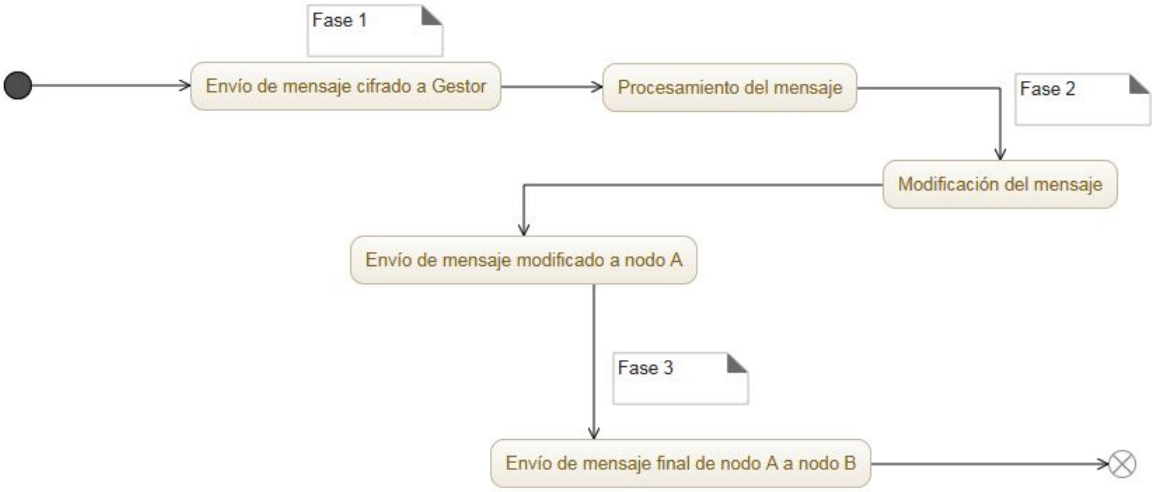

Figura 8. Diagrama de actividad del envío de mensajes

Sistema Experto

En este apartado se van a definir las reglas y acciones que definen el funcionamiento del sistema experto.

El sistema experto se ha dividido en dos elementos principales, uno destinado a la autenticación del nodo emisor, para el cual se determinarán el número limitado de acciones que se le permite realizar dentro del sistema; otro destinado a determinar qué tipo de modificaciones han de realizarse cuando el sistema recibe un mensaje.

La estructura de estos dos elementos se basa en tres atributos fundamentales que han de ser registrados previamente al uso del gestor:

Fase de ingreso al sistema.

<**sujeto**> := $<ID_s><rol_s>$

Tales que:
$<ID_s>$:= Nombre del nodo emisor.
$<rol_s>$:= Admin | Engineer | Node (ver tabla 7)

Fase de modificación del mensaje

<**sujeto**> := $<ID_s><tipo_s><cifrados_s>$

Tal que:
$<ID_s>$:= Nombre del nodo emisor.
$<tipo_s>$:= H | W | P (ver Tabla 8)
$<cifrados_s>$:= Cifrados que soporta el nodo emisor. (ver sección 7.3)

<**objeto**> := $<ID_o><tipo_o><cifrados_o>$
El objeto adquiere los mismos atributos que el nodo sujeto.

<**mensaje**> := $<ID_s><tipo_s><cifrados_s>$

Tras la definición genérica del sistema experto, se explicarán en detalle las distintas acciones que realiza el gestor cuando inicia una comunicación.

6.1. RBAC para el acceso al sistema

Se define RBAC (Role Based Access Control en inglés) como el control de acceso basado en roles [21] determinando qué se le permite a los usuarios dados sus roles. En el caso de este proyecto, se ha utilizado este modelo para diferenciar tres tipos de usuarios: administrador, ingeniero y nodo.

Administrador: este es el rol más completo. El administrador del sistema tiene acceso a todo el contenido del gestor, y es el que mantiene toda la gestión de usuarios. Este rol tiene las siguientes capacidades:

- Imprimir el estado del Log del sistema.
- Imprimir el estado de la base de datos de usuarios registrados.
- Generar un nuevo par de claves para el sistema.
- Insertar un nuevo usuario.
- Borrar un usuario.

En el caso de este rol es importante destacar que, cuando se realiza una generación de nuevas claves, el sistema comienza un proceso de recifrado de las tablas destinadas al almacenamiento de los datos de usuarios. Esto se realiza con el fin de mantener la integridad de los datos cuando se actualiza dichas claves.

Ingeniero: un usuario con el rol de ingeniero se encarga de la monitorización de las comunicaciones. Por tanto, este usuario tiene únicamente acceso al estado del Log del sistema, además de ciertas opciones comunes al resto de usuarios, como son:

- Utilizar el sistema gestor de políticas con un mensaje dado.
- Generar un nuevo par de claves.

Nodo: este rol es el destinado a la mayoría de dispositivos que utilizarán el sistema. Este rol únicamente tiene disponibles las opciones comunes a los tres roles, ya que es supuesto que este rol está destinado al uso propio del gestor para la comunicación.

En el sistema únicamente puede haber registrado un solo administrador, ya que la existencia de un número mayor de ellos sería perjudicial para la integridad de los datos almacenados.

Tabla 7: Roles de usuarios

Rol	Escribir	Leer	Utilizar el sist.
Admin	✓	✓	✓
Eng		✓	✓
Node			✓

6.2. Acciones del sistema experto

En esta sección se van a tratar todos los casos posibles y sus acciones. Por simplicidad, se definen dos nodos que quieran comunicarse utilizando criptografía simétrica como A y B, siendo A el nodo emisor, y B el nodo receptor. Además, se tendrán en cuenta los tipos de nodos que están disponibles en el sistema. Estos son: W (débil), P (fuerte), H (gran potencia) (ver tabla 8).

6.2.1. Caso tipo A igual a tipo B

Este caso abarca todos aquellos escenarios en los que ambos dispositivos tengan las mismas características a nivel computacional. Sin embargo, es posible que esta comunicación no sea posible dependiendo del catálogo de cifrados que posean. Por

ello, se evalúan dichos catálogos y se busca la opción que haga posible la comunicación entre ellos. En el caso de no existir compatibilidad, el gestor finaliza su ejecución haciéndole saber al usuario que no es posible modificar su mensaje. Si por el contrario se ha encontrado un cifrado compatible, el gestor procede a modificar dicho mensaje para adaptarlo a las posibilidades del nodo B, receptor de la comunicación.

6.2.2. Caso tipo A mayor a tipo B

Este caso analiza las capacidades del nodo B para que le sea posible la recepción del mensaje. En este escenario, el dispositivo A es capaz de utilizar prácticamente cualquier tipo de cifrado, pero puede existir la posibilidad de que el nodo B no sea capaz de leer el mensaje directamente. Es así que el gestor, sirviéndose del sistema experto, modifica su mensaje de la misma forma que hace en el caso anterior.

6.2.3. Caso tipo A menor a tipo B

Este caso es el que menor carga computacional tiene, puesto que este escenario abarca aquellas situaciones en las que un dispositivo de baja potencia se comunica con otro que tiene capacidades superiores a este, y en este caso la probabilidad de que un dispositivo receptor no pueda computar con un cifrado. De forma similar, si no es posible la comunicación directa, se procede a la modificación del mensaje si es posible.

Tabla 8: Tipos de usuarios por capacidades

Tipo	**Valor**
W	Dispositivo de capacidad baja. Ej. IoT, móviles.
P	Dispositivos de capacidad media. Ej, portátiles, PC.
H	Dispositivos de capacidad alta. Ej, servidores, centros de alta computación.

En cada caso, el acceso a los datos de los usuarios que utilizan el sistema está protegido por un cifrado asimétrico el cual únicamente es recuperable por el servidor que aloja el sistema. En el proceso de evaluación de reglas se descifran dichos datos y se utilizan para determinar la acción correspondiente. Esto reduciría el daño producido

por un ataque directo a la base de datos, reforzando así la seguridad del sistema. En la sección 7.3 se detallan las especificaciones de estos cifrados, el cual se utiliza en todas las comunicaciones.

Desarrollo de la aplicación Android

Para una correcta evaluación del sistema en funcionamiento, se propuso el desarrollo de una aplicación para sistemas Android que permitiera simular un paso de mensajes utilizando el protocolo TCP/IP. (Transmission Control Protocol / Internet Protocol, en inglés)

Android es un sistema operativo para dispositivos portables, como pueden ser teléfonos móviles, tablets, relojes inteligentes e incluso televisores y automóviles. Ha sido desarrollado por Google y está basado en el Kernel de Linux [12]. Es uno de los sistemas operativos más popularizados en este sector debido a la proliferación de dispositivos móviles que se ha producido a lo largo de los últimos años. Para el desarrollo de la aplicación se ha utilizado el lenguaje de programación Java, ayudándose del entorno de desarrollo integrado o IDE (Integrated Development Environment, en inglés) Android Studio, que permite, de forma sencilla, el diseño y desarrollo de aplicaciones para este tipo de dispositivos.

TCP hace referencia a *Transmission Control Protocol* o Protocolo de control de transmisión. El objetivo de este protocolo es realizar una conexión segura, gracias a la existencia de un valor de reconocimiento, o *ACK* (de *Acknowledgment*, en inglés), el cual permite saber en cada comunicación si se ha recibido un mensaje correctamente. Esto hace que el envío de los mensajes propuestos en este proyecto sea más seguro. Además, este protocolo actúa en la capa de transporte (ver figura 9),y añade al protocolo de Internet, IP (Internet Protocol, en inglés), las funciones necesarias para permitir a la comunicación realizada por dos sistemas sea libre de errores, sin pérdidas y con seguridad [8].

Figura 9. Pila de protocolos

Un *socket* es un punto de origen o de destino entre dos procesos que se ejecutan en terminales diferentes [16]. Existen dos tipos básicos de socket: TCP y UDP (User Datagram Protocol por sus siglas en inglés, o Protocolo de Datagramas de Usuario). Debido al propósito del proyecto de realizar una conexión para enviar información, se ha empleado el protocolo TCP, ya que el protocolo UDP no está orientado a la conexión.

Las características de este protocolo son, entre otras [16]:

- Es un protocolo orientado a la conexión.
- Ofrece un servicio fiable.
- Conexión orientada a flujo de bytes punto a punto.

El esquema básico de programación de este protocolo es el siguiente:

Acciones del servidor

- Crear un socket.
- Esperar una conexión.
- Aceptar la conexión.
- Enviar/recibir bytes de información.
- Cerrar la conexión.

Acciones del cliente

- Crear un socket para conectarse al servidor.
- Enviar/recibir bytes de información.
- Cerrar la conexión.

7.1. Fases de actuación

Al ser una conexión orientada al flujo de bytes, en cada envío y recepción de mensajes, cada parte deberá indicar el tamaño en bytes del mensaje a procesar. En el caso de la aplicación diseñada en este proyecto, se han definido dos fases principales, siguiendo la figura 7:

En la primera fase, el terminal móvil envía al gestor, servidor del modelo, un mensaje que desea hacer llegar a un terminal receptor que procesará y actuará en función de dicho mensaje. Para poder iniciar la conexión con el *socket* del servidor, el usuario de la aplicación deberá indicar en los espacios dedicados para ello, la dirección IP del servidor y el puerto que dicho servidor haya abierto. Una vez introducidos, el usuario podría enviar el mensaje.

En este momento, el terminal se encuentra en modo espera a que el servidor envíe una respuesta. El gestor recibe el mensaje y lo procesa, haciendo las modificaciones necesarias, enviándole el nuevo mensaje de vuelta al teléfono, y cerrando la conexión entre el terminal móvil y el gestor.

En la segunda fase, la aplicación muestra el mensaje transformado con el cifrado del nodo receptor, permitiéndole al usuario enviárselo haciendo uso de otra conexión TCP. El proceso es idéntico al de la primera fase, con la diferencia de que no se espera ningún mensaje tras este envío. Una vez repetido el proceso de envío de mensaje, el nodo B o nodo receptor habrá recibido el mensaje deseado.

Microcontrolador Arduino

Debido al avance tecnológico, es muy común encontrarse en un hogar al menos uno de los llamados dispositivos que pertenecen a las *Smart Things* (Cosas inteligentes). Estos dispositivos, para su correcto funcionamiento y para ofrecer una mejor experiencia de usuario, se encuentran conectados a internet de forma continua, exponiéndolos a ataques que podrían comprometer la privacidad y seguridad de sus usuarios. Es por esto que se ha integrado al proyecto un microcontrolador Arduino que será representante de estas nuevas tecnologías.

Arduino es un microcontrolador *open-source* diseñado para el prototipado rápido, es de fácil uso y se creó con el propósito de vincular software y hardware. Es de bajo coste y permite su uso en múltiples plataformas (Windows, Mac, Linux), lo cual lo hace integrable en cualquier escenario [2].

Se ha elegido utilizar este dispositivo para poder demostrar la utilidad del sistema en un entorno seguro, haciendo uso de las tecnologías mencionadas en este proyecto.

Este dispositivo se denomina *Single-Board Computer*, debido a que sus componentes se encuentran integrados en una única placa ampliable. Gracias a esto, es posible realizar un pequeño escenario en el que se encenderá una bombilla LED (ver figura 10) en función de la instrucción recibida por la comunicación realizada entre el dispositivo móvil y Arduino (ver capítulo 9).

El esquema del circuito es el siguiente:

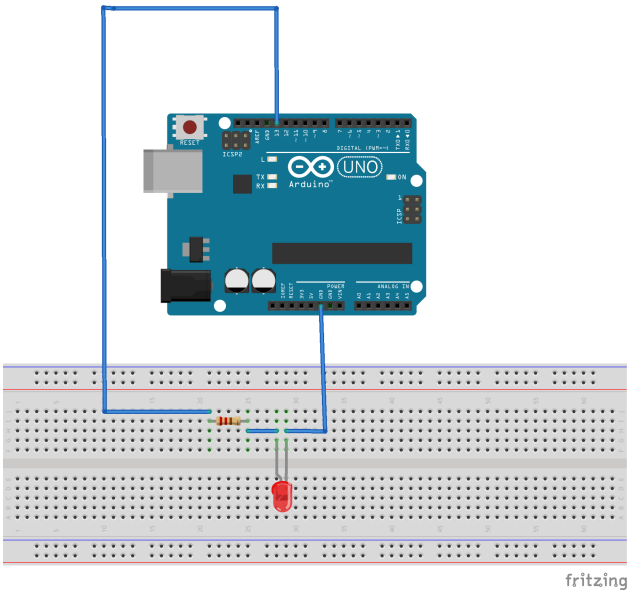

Figura 10. Esquema del circuito

Para el funcionamiento del Arduino se ha utilizado el lenguaje de programación C para realizar una conexión con un ordenador portátil, seguido de un pequeño programa en Python que hará de punto B para la aplicación móvil.

El programa en Python está diseñado para leer la información de un fichero que se obtendrá del envío de información por parte de la aplicación móvil. Tras el procesamiento de dicho fichero, el Arduino descifrará y aplicará la acción de encender el LED mostrado en el circuito un número determinado de veces, que vendrá fijado por dicho mensaje.

Pruebas y resultados

Con el fin de que la funcionalidad del sistema y de sus elementos principales cumplan los requisitos y se adapten correctamente al medio de ejecución, se han realizado una serie de pruebas que se expondrán en este apartado.

9.1. Registro de usuario

Cuando un usuario desea utilizar la aplicación por primera vez, se le mostrará un diálogo (ver figura 11).

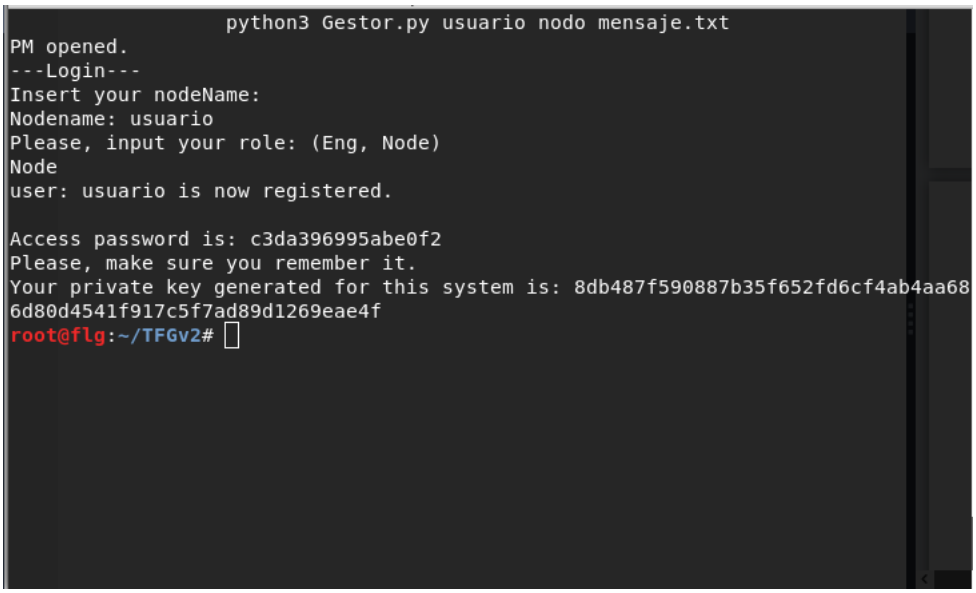

Figura 11. Diálogo de registro

Se puede observar como el sistema registra al usuario correctamente, mostrándole por pantalla su contraseña de acceso y su clave privada para el uso de cifrado con clave pública. Esta acción se realiza una sola vez y se entiende que se hace en un entorno seguro, por conveniencia de uso en este proyecto.

9.2. Inserción de un usuario

Un administrador es capaz de insertar un usuario aportando los datos necesarios que dan identidad al mismo. Esta acción es posible haciendo uso de las opciones que se le dan a este tipo de usuario (ver figuras 12 y 13).

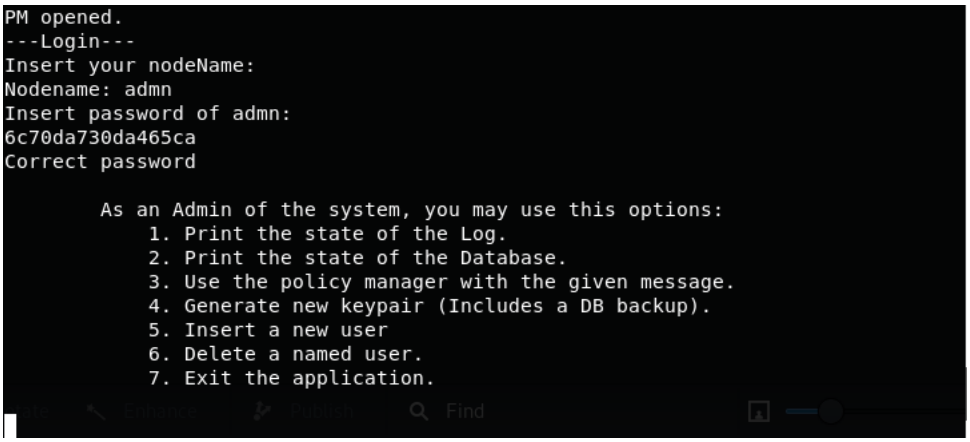

Figura 12. Opciones del administrador

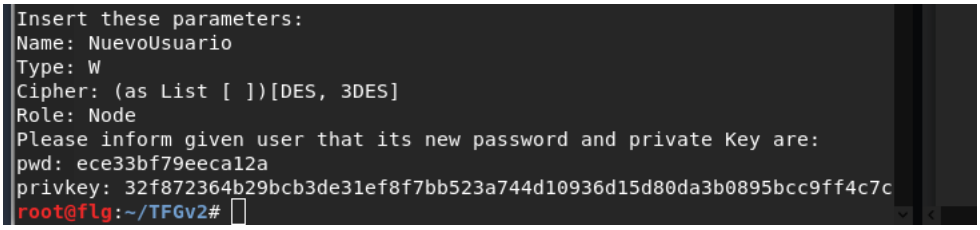

Figura 13. Inserción de un usuario nuevo por parte del administrador

9.3. Cambio de claves ECC

Si un usuario lo desea, puede modificar sus claves pública y privada. Si se utiliza esta opción, se mostrarán por pantalla el par de claves (ver figura 14), y se almacenará en la correspondiente tabla de la base de datos la clave pública para el uso del sistema (ver figura 15).

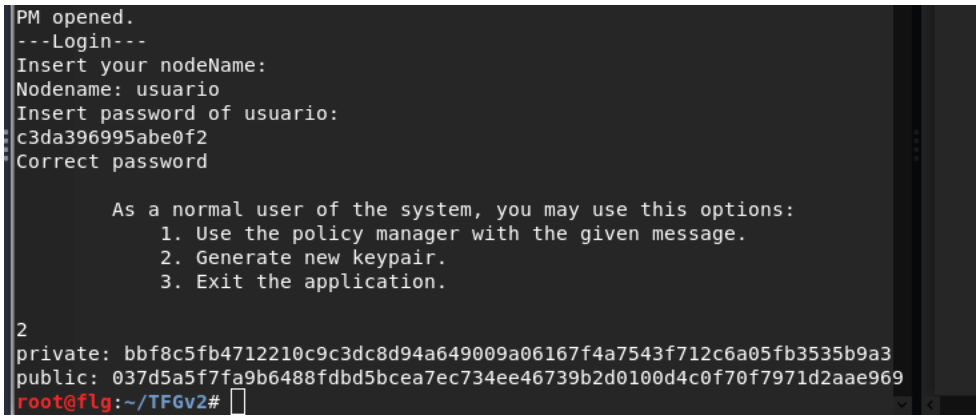

Figura 14. Cambio de claves por el usuario

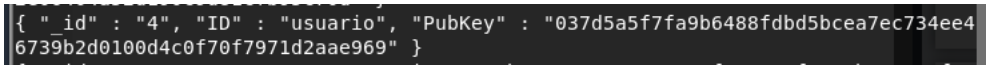

Figura 15. Modificación en la base de datos de las claves

9.4. Eliminación de usuario

Un administrador podrá eliminar un usuario aportando su nombre usado en el registro. En el momento en el que el usuario vuelva a acceder al sistema, este deberá aportar nuevos datos para su registro (ver figura 16).

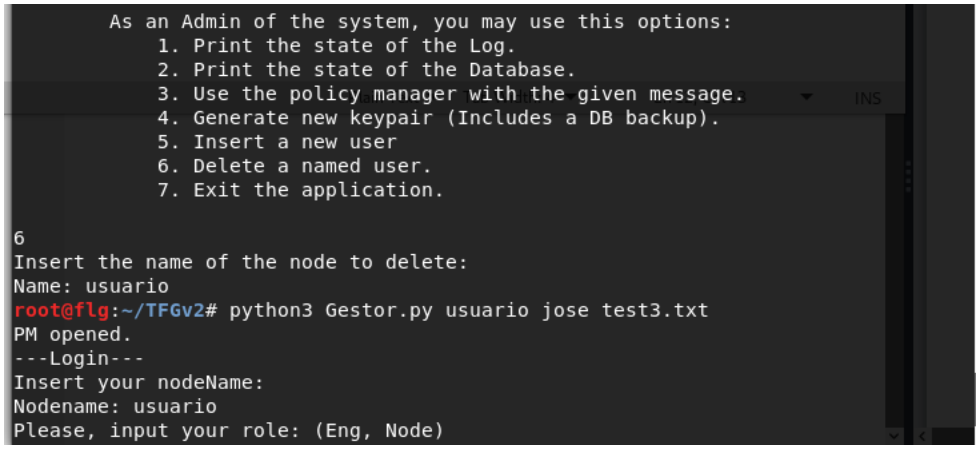

Figura 16. Eliminación de usuario

9.5. Muestra del log de comunicaciones

El administrador o un ingeniero habilitado podrá acceder al registro del sistema. En él únicamente se muestra la información relativa al nodo emisor y al nodo receptor, junto a la fecha en la que se realizó la comunicación (ver figura 17).

Figura 17. Muestra de un log de comunicación

9.6. Envío de mensajes

Cuando dos usuarios desean realizar un envío de mensajes, se le proporcionará al sistema gestor de políticas el mensaje correspondiente con su formato establecido. Para esta prueba se ha querido enviar un mensaje a un dispositivo Arduino en el que se le dan instrucciones de hacer parpadear un LED 5 veces. Como modo de cifrado se ha utilizado AES_EAX y se han creado una clave y un *Nonce* adecuado para ello.

En la aplicación Android se deberá insertar la IP y el puerto por el que se enviará el mensaje (ver figura 18). Tras este paso, se encarga del envío de este mensaje al sistema, con su correspondiente respuesta (ver figura 19). El usuario entonces utilizaría el botón *Send Arduino* para enviar dicho mensaje al nodo receptor.

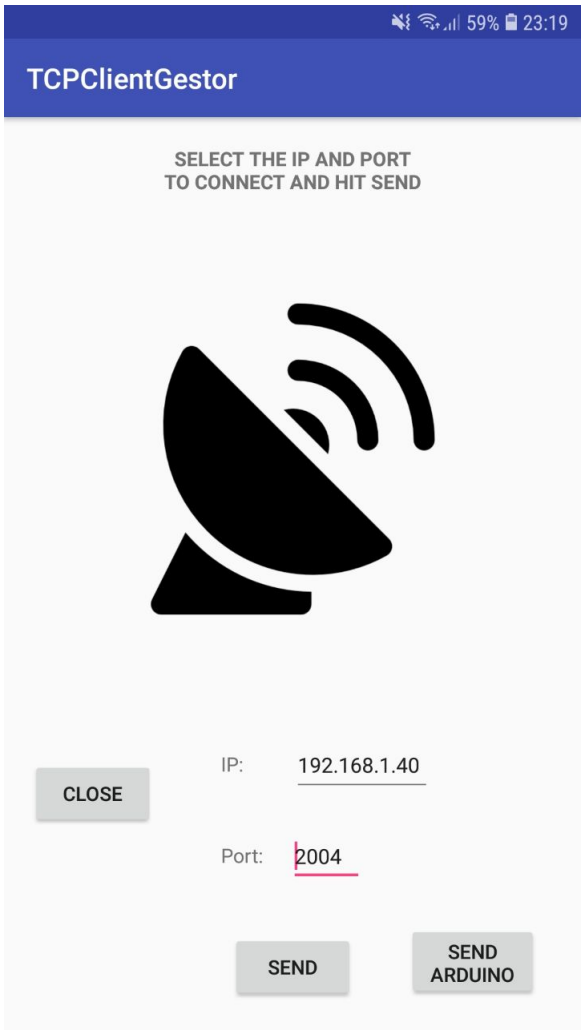

Figura 18. Ventana inicial de la aplicación Android

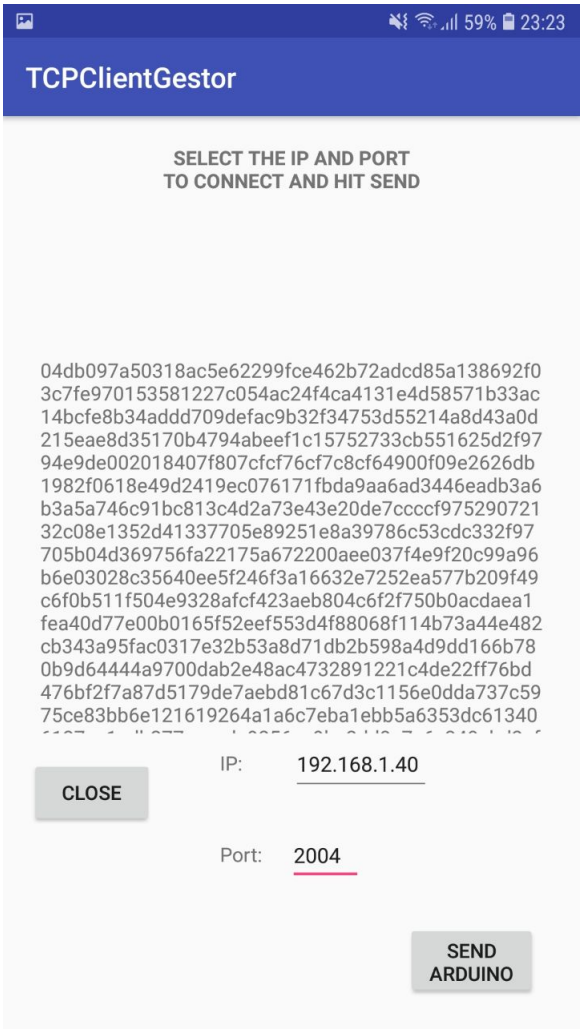

Figura 19. Recepción del mensaje por parte del sistema

Por otro lado, el sistema Arduino recibirá el mensaje, y mostrará por pantalla su acción, seguido del parpadeo del LED (ver figura 20).

Figura 20. Recepción del mensaje del Arduino

En el caso de que el mensaje recibido haya sido modificado por un ataque en el envío, es posible que no se pueda verificar la integridad del mismo. Por tanto, si esto ocurre, se deberá comprobar si el *hash* del mensaje recibido coincide con el enviado

(ver figura 21).

Figura 21. Fallo en la integridad del mensaje

9.7. Cumplimiento de requisitos

Tras la realización de las pruebas, en este apartado se mostrarán los requisitos obtenidos tras su análisis y su cumplimiento de acuerdo de dichas pruebas (ver tabla 9). Se valorarán de acuerdo a tres categorías. Estas categorías miden el grado de satisfacción, y son:

- **Alto**: el requisito se ha cumplido completamente y sin fallos.
- **Medio**: el requisito se ha cumplido pero no cumple todas las expectativas.
- **Bajo**: el requisito no se cumple con los mínimos objetivos.

Tabla 9: Cumplimiento de requisitos

Requisito	Probado	Satisfacción
RF01	✓	Alto. Cualquier dispositivo es capaz de registrarse sin problemas.
RF02	✓	Alto. Tras cambiar las claves, todos los campos se modifican correctamente.
RF03	✓	Alto. Es el objetivo principal, y el gestor realiza la acción con normalidad.
RF04	✓	Alto. El registro se muestra correctamente, y únicamente al nodo habilitado.
RF05	✓	Alto. El nodo habilitado (Admin) es capaz de insertar un usuario correctamente.
RF06	✓	Alto. El nodo habilitado (Admin) es capaz de eliminar un usuario aportando su nombre.
RNF01	✓	Alto. Las contraseñas se encuentran protegidas por hash en todo momento desde su creación hasta su modificación.
RNF02	✓	Medio. La base de datos está cifrada. Sin embargo, no todas las tablas se han cifrado, únicamente aquella tabla que contiene datos sensibles al usuario.
RNF03	✓	Alto. En todo momento los mensajes se transmiten de forma cifrada con clave pública usando ECC.
RNF04		Medio. Únicamente los nodos administradores pueden obtener la información relativa a la base de datos.

Trabajo futuro

Tras el tiempo dedicado al desarrollo de este TFG, se han observado una serie de mejoras que podrían aplicarse a este sistema. A continuación, se listan algunas de ellas:

- **Inclusión de una interfaz de usuario**: aunque este sistema está diseñado para ser un elemento transparente al usuario, es posible que en algunos casos sea interesante la utilización de una interfaz para la administración del mismo.
- **Ampliación de métodos de cifrado**: en este proyecto se han incluido un catálogo limitado de cifrados, sin embargo a medida que van mejorando las tecnologías es posible una ampliación de este para mejorar la funcionalidad del sistema.
- **Automatización**: se han considerados algunos aspectos para que se realicen de forma automática, sin embargo una mejora a considerar podría ser la automatización en el proceso de registro y autenticación, que podría ser obteniendo un *token* de autenticación por parte del usuario que sirva para determinar los parámetros utilizados en el momento del registro.
- **Base de datos en la nube**: el sistema pretende ser un elemento universal que pueda ser usado en un entorno sin restricciones y disponible en cualquier momento. Por ello, almacenar la base de datos en un servicio *Cloud* puede ser una mejora del sistema desarrollado.

Conclusiones

En un momento tan crítico como es el actual para la seguridad de la información, es de gran importancia tener en cuenta todos los aspectos que conciernen a todas las tecnologías. Con este proyecto, se ha propuesto cubrir una necesidad clave como es la seguridad en las comunicaciones de los dispositivos más comunes. En el momento de realización de este proyecto, el número de dispositivos inteligentes en los hogares aumenta exponencialmente, haciendo vulnerables a miles de personas si no se le aplica la debida protección.

Este proyecto ha demostrado ayudar en este proceso de protección, ofreciendo una capa de seguridad que hará más difícil el robo de información no deseada y mejorará la vida de las personas.

Desarrollar este sistema ha supuesto una madurez en conceptos relacionados con la seguridad informática, como son las redes o la criptografía, y también se han sentado bases respecto al desarrollo de código seguro. Gracias a la versatilidad del lenguaje Python y de sus aplicaciones en el campo de la seguridad, este proyecto ha sido posible.

El objetivo final de este proyecto es que sea utilizado por el mayor número de personas, o que sirva de iniciativa para aplicar mejoras de seguridad en los sistemas en los que la sociedad se encuentra envuelta con la llegada de las *Smart Cities*.

Bibliografía

[1] *Amazon alexa*. https://developer.amazon.com/es/alexa.

[2] *Arduino*. https://www.arduino.cc/en/Guide/Introduction.

[3] *Authenticated encryption*. https://en.wikipedia.org/wiki/Authenticated_encryption.

[4] *F-security bussines policy manager*. https://www.f-secure.com/en/business/products.

[5] *National institute of standards and technology*. https://www.nist.gov/.

[6] *Pycryptodome*. https://pycryptodome.readthedocs.io.

[7] *Pyknow: Expert systems for python*. https://pyknow.readthedocs.io/en/stable/.

[8] *Redes informáticas/protocolos tcp y udp en el nivel de transporte*. https://es.wikibooks.org/wiki/Redes_informáticas/Protocolos_TCP_y_UDP_en_el_nivel_de_transporte.

[9] *Salted password hashing - doing it right*. https://crackstation.net/hashing-security.htm.

[10] *Security and privacy challenges in the internet of things*. https://journal.ub.tu-berlin.de/index.php/eceasst/article/viewFile/208/205.

[11] *Security model in iot*. https://affinity-it-security.com/security-model-iot-devices/.

[12] *Sistema operativo android*. https://es.wikipedia.org/wiki/Android.

[13] *Usgov security policies for bussiness*. https://www.business.gov.au/risk-management/cyber-security/creating-a-cyber-security-policy-for-your-business.

[14] *¿qsué es scrum?* https://proyectosagiles.org/que-es-scrum/.

[15] N. Aidinyantz, *A glossary of cryptographic algorithms.* https://www.globalsign.com/en/blog/glossary-of-cryptographic-algorithms/.

[16] M. Amor Pinilla, I. Ayala Viñas, and J. M. Horcas Aguilera, *Servicios básicos para el nivel de transporte en internet.* Campus Virtual - Redes y Sistemas Distribuidos.

[17] A. Ardiri, *Is it possible to secure micro-controllers used within iot? — evothings.* https://evothings.com/is-it-possible-to-secure-micro-controllers-used-within-iot/.

[18] E. B. Barker, *Digital signature standard (dss).* https://www.nist.gov/publications/digital-signature-standard-dss-2.

[19] ——, *Recommendation for key management.* https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf#page=37.

[20] M. Brennan Jones, *Cpu info for python.* https://github.com/workhorsy/py-cpuinfo.

[21] S. Carter, *Rbac vs abac access control models - iam explained.* https://blog.identityautomation.com/rbac-vs-abac-access-control-models-iam-explained.

[22] A. García, *El cifrado sha-1 ya no es seguro.* https://www.adslzone.net/2017/02/23/cifrado-sha-1-ya-no-seguro-google-lo-ha-roto-despues-22-anos/.

[23] S. Gnana and A. Asir, *Performance analysis of data encryption algorithms for secure data transmission.* https://www.researchgate.net/publication/316789478_Performance_Analysis_of_Data_Encryption_Algorithms_for_Secure_Data_Transmission.

[24] G. Keizer, *Microsoft sets 'obsolete' password reset practice.* https://www.computerworld.com/article/3391365/microsoft-tells-it-admins-to-nix-obsolete-password-reset-practice.html.

[25] M. R. Khalifeh Soltanian and I. S. Amiri, *Birthday attack.* https://www.sciencedirect.com/topics/computer-science/birthday-attack.

[26] J. Lancrenon, D. Khader, P. Y.A.Ryan, and FengHao, *Key exchange protocol.* https://www.sciencedirect.com/science/article/pii/B9780128038437000491.

[27] W. Li, *Eciespy - elliptic curve integrated scheme for secp256k1 in python.* https://github.com/kigawas/eciespy.

[28] C. Pollack, *Password guidelines.* https://www.fpainc.com/blog/password-guidelines-from-nist.

[29] Silabs, *Iot security part 7: Key exchange using elliptical curve cryptography.* https://www.silabs.com/community/blog.entry.html/2016/05/26/iot_security_part7-UeMh.

[30] N. Sullivan, *A (relatively easy to understand) primer on elliptic curve cryptography.* https://arstechnica.com/information-technology/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/.

[31] S. Turner, D. Brown, K. Yiu, R. Housley, and T. Polk, *Elliptic curve cryptography subject public key information.* https://tools.ietf.org/html/rfc5480.

Anexo A: Lista de acrónimos

- AES: Advanced Encryption Standard - Estándar de cifrado avanzado.
- ACK: Acknowledgment - Reconocimiento (Redes).
- CBC: Code Block Chaining - Cifrado por bloques.
- DES: Data Encryption Standard - Estándar de cifrado de datos.
- EAX: Encrypt-then-Authenticate-then-Translate - Modo de cifrado autenticado.
- ECC: Elliptic Curve Criptography - Criptografía de curvas elípticas.
- IoT: Internet of Things - Internet de las cosas.
- IP: Internet Protocol - Protocolo de internet.
- IV: Initialization Vector - Vector de inicialización.
- LED: Light Emitting Diode - Diodo emisor de luz.
- NIST: National Institute of Standards and Technology - Instituto nacional de estándares y tecnología de los Estados Unidos.
- Nonce: Number only used once - Número usado una vez (Redes).
- RBAC: Role Based Access Control - Control de acceso basado en roles.
- RSA: Rivest, Shamir y Adleman. Criptografía de clave pública que obtuvo su nombre por sus autores.
- SHA: Secure Hash Algorithm - Algoritmo seguro de hash.
- TCP: Transmission Control Protocol - Protocolo de control de transmisión.
- UDP: User Datagram Protocol - Protocolo de datagramas de usuario.

Anexo B: Manual de usuario

B.0.1. Instalación de la aplicación Android

Para instalar la aplicación, será necesario trasladar el archivo APK al dispositivo a utilizar. Esta instalación deberá hacerse tras dar los permisos necesarios al teléfono respecto a la instalación de aplicaciones de origen desconocido. Este permiso se encuentra normalmente en Ajustes >Seguridad >Orígenes desconocidos.

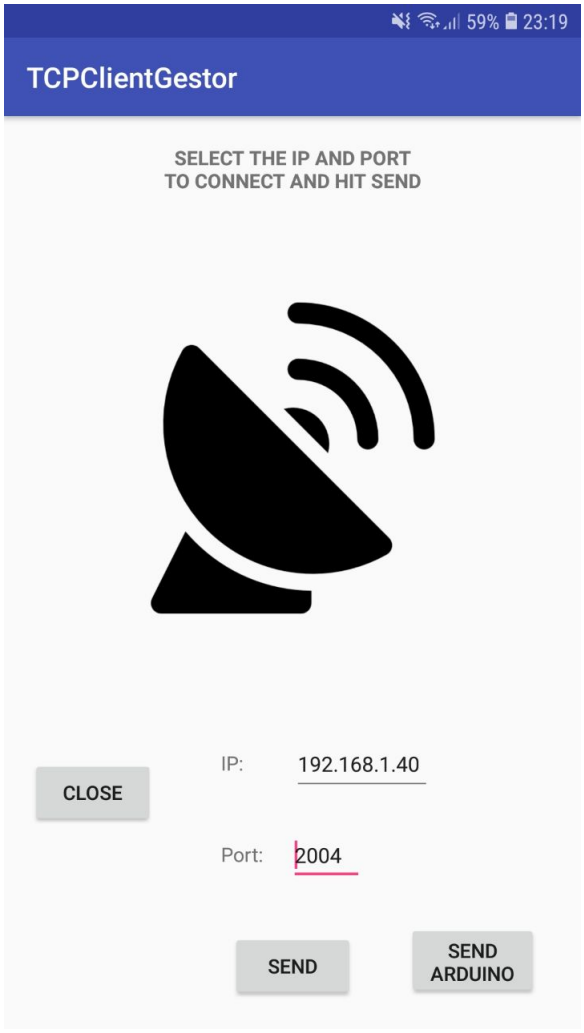

Figura 22. Ventana inicial de la aplicación Android

Como se muestra en la figura 22, en esta ventana se permite la inserción de una dirección IP y puerto de conexión. Estos son los utilizados para enviar el mensaje al sistema (servidor). Dicho mensaje se encuentra en la carpeta Descargas del dispositivo con el nombre **message.txt**. Este mensaje contiene el cifrado con clave pública del servidor del contenido original que se desea enviar.

Respecto a los botones que se muestran en (figura 22):

- **Close**: cierra la conexión con el socket correspondiente.
- **Send**: envía el mensaje descrito anteriormente al servidor.
- **Send Arduino**: envía la respuesta del mensaje al microcontrolador Arduino.

B.0.2. Instalación de Arduino

Para el uso del microcontrolador Arduino, será necesario tener instalado el entorno de desarrollo obteniéndolo del siguiente enlace:

https://www.arduino.cc/en/Main/Software

Tras su instalación, será necesario instalar en el propio Arduino el código que permitirá conectarse con el ordenador vía USB. Este *Sketch* se encuentra en las opciones: File >Examples >Firmata >StandardFirmata. Este programa deberá ser cargado al Arduino, y se podrá utilizar el programa facilitado que permitirá la conexión con la aplicación Android, utilizando los siguientes comandos en un terminal:

>_ ./Arduino.py

Será necesario modificar el campo *host* dentro del código fuente para que se adecue a la red en la que se esté utilizando.

B.0.3. Instalación del sistema gestor de políticas

Para la instalación del sistema, se deberá tener instalada una versión de MongoDB para el uso de la base de datos. Esta base de datos podrá encontrarse siguiendo el siguiente enlace: https://www.mongodb.com/download-center/community

Para el uso en sistemas Windows, es recomendable el uso de la herramienta MongoDB Compass (https://www.mongodb.com/download-center/compass), ya que gracias a su interfaz de usuario, es posible una fácil gestión de la base de datos.

Si se utilizan sistemas Linux, se deberá instalar siguiendo como recomendación la documentación propia de Mongo (https://docs.mongodb.com/manual/administration/install-on-linux/). Tras esto, se proporciona un script para el inicio de la base de datos. Para utilizarlo, será necesario abrir un terminal, e introducir los siguientes comandos:

```
>_ chmod +x Mongod.sh
>_ ./Mongod.sh
```

Para una gestión de la base de datos, se deberá iniciar el proceso mongo dentro de la carpeta de instalación de MongoDB.

Si no se tiene instalado Python3, se recomienda al lector seguir las siguientes instrucciones de la documentación oficial de Python (https://www.python.org/downloads/).

El gestor de políticas requiere del uso de las siguientes librerías de Python3 (junto a su comando de instalación):

- **eciespy**: pip install eciespy.
- **pyknow**: pip install pyknow.
- **cryptodome**: pip install pycryptodomex / pip install pycryptodome.

B.0.4. Uso del gestor de políticas

Para la conexión con la aplicación Android, deberá lanzarse el siguiente comando en un terminal:

>_ ./Server.py

Será necesario modificar el campo *host* dentro del código fuente para que se adecue a la red en la que se esté utilizando.

Para utilizar el gestor, se deberá lanzar por terminal el siguiente comando Python:

>_ python3 Gestor.py [NodoA] [NodoB] [Mensaje]

El sistema escribe los resultados imprimidos por terminal a un fichero llamado *msg.txt*. En el caso del uso del servidor, este recibe la información de la aplicación Android en un fichero llamado *Received.txt*

En caso de duda al usar el sistema, se ofrece un pequeño manual de ejecución al utilizar la opción **-h** al invocar el proceso, y sin ninguno de los otros parámetros.